2018-2019年度第二学期 00106501

计算机图形学

童 伟 华　管理科研楼1205室

E-mail: tongwh@ustc.edu.cn

中国科学技术大学　数学科学学院

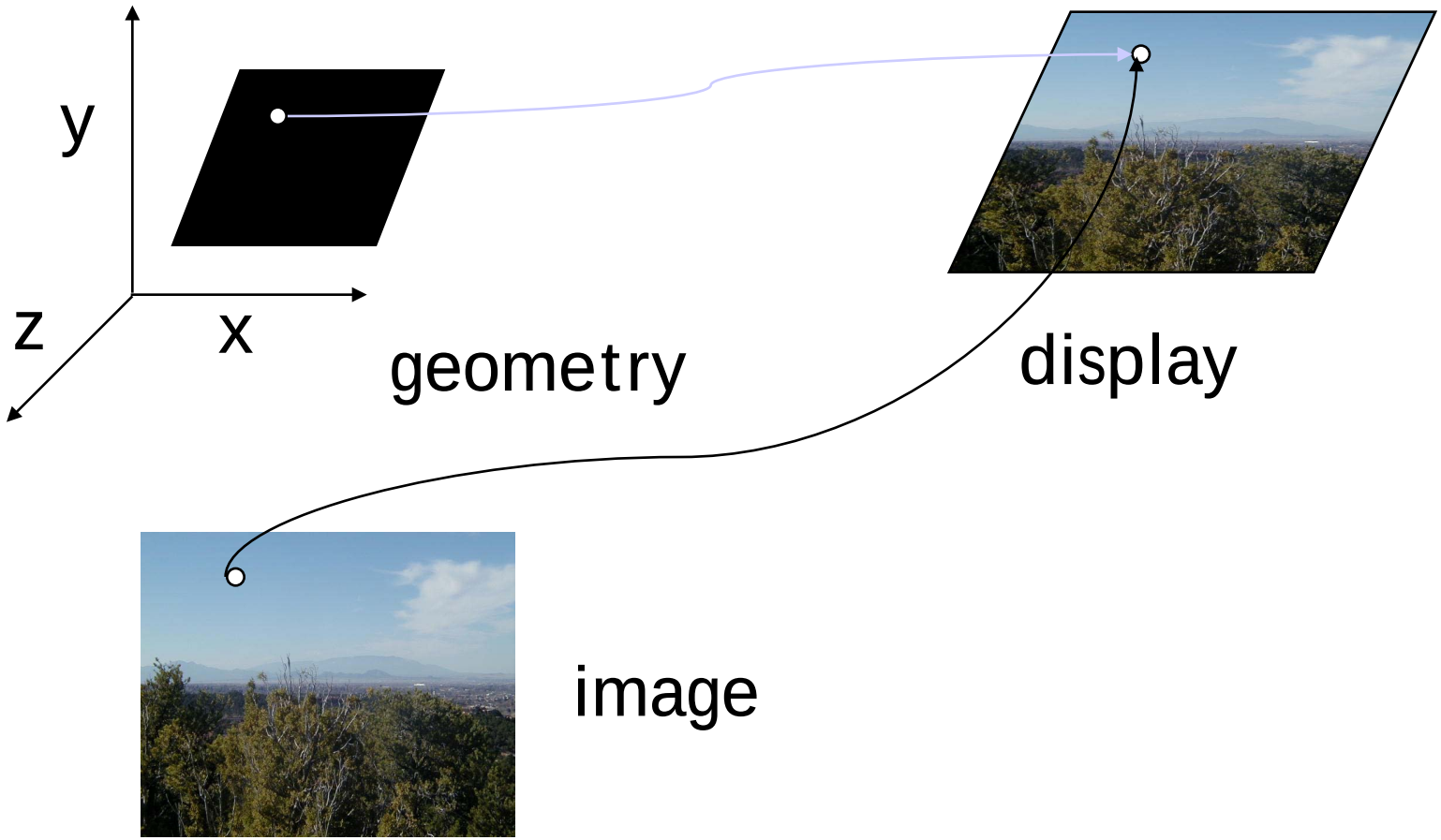
http://math.ustc.edu.cn/

第三节 OpenGL中的纹理映射

OpenGL与纹理

- OpenGL支持的纹理类型
  - 一维、二维、三维纹理
  - 一维、二维纹理数组
  - 二维多重采样纹理/数组
  - 立方体映射/数组
  - 二维矩形纹理（没有mipmap，不支持纹理数组）
  - 纹理缓存（一维缓存）
- 纹理通过纹理单元绑定到OpenGL环境，纹理单元的命名：GL_TEXTURE0 - GL_TEXTUREi
- 在着色器中，可以使用采样器变量来访问纹理

纹理映射



应用纹理映射的步骤

- 创建一个纹理对象，加载纹理数据
- 确定纹理如何应用到每个像素上（设置滤波函数，纹理函数，wrap模式，透视校正等参数）
- 启用纹理映射功能，将纹理单元与着色器中将要使用的纹理采样器关联
- 绘制场景，提供纹理坐标和几何图形坐标
- 在着色器中使用纹理采样器来查询纹素值

- 下面以二维纹理映射为例

创建和初始化纹理

- 创建纹理对象：glGenTextures
- 绑定纹理对象：glBindTexture
- 激活纹理单元：glActiveTexture

- 判断一个对象是否为纹理对象： glIsTexture
- 删除纹理对象：glDeleteTextures

指定纹理对象

- 第一种方式：设置纹理格式+数据
  - void glTexImage2D(GLenum target, GLint level, GLint internalFormat, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const void *data);

- 第二种方式：设置纹理格式，替换纹理数据
  - void glTexStorage2D(GLenum target, GLsizei levels, GLenum internalFormat, GLsizei width, GLsizei height);
  - void glTexSubImage2D(GLenum target, GLint level, GLint xoffset, GLint yoffset, GLsizei width, GLsizei height, GLenum format, GLenum type, const void *data);

加载纹理数据

- 第一种方式：显示设置纹理数据
  - 直接设置数据，缺省的布局：从左到右，从顶到底
- 第二种方式：使用Pixel unpack缓存
  - `glBindBuffer(GL_PIXEL_UNPACK_BUFFER, buf)`
  - …
- 第三种方式：从帧缓存拷贝数据
  - `glCopyTexImage2D / glCopyTexSubImage2D`
- 第四种方式：从文件加载图像
  - 使用辅助的库，譬如stb_image.h

设置纹理参数

- 第一种方式：使用纹理对象内置的采样器
  - glTexParameter{fi}/glTexParameter{fi}v/glTexParameterI{i ui}v
  - 设置Warp方式，譬如：glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
  - 设置过滤函数，譬如：glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

- 第二种方式：使用采样器对象
  - 生成、绑定、判断、删除采样器等：glGenSamplers, glBindSampler, glIsSampler, glDeleteSamplers等
  - glSamplerParameter{fi} / glSamplerParameter{fi}v/glSamplerParameterI{i ui}v
  - 设置纹理参数，与glTexParameter函数类似

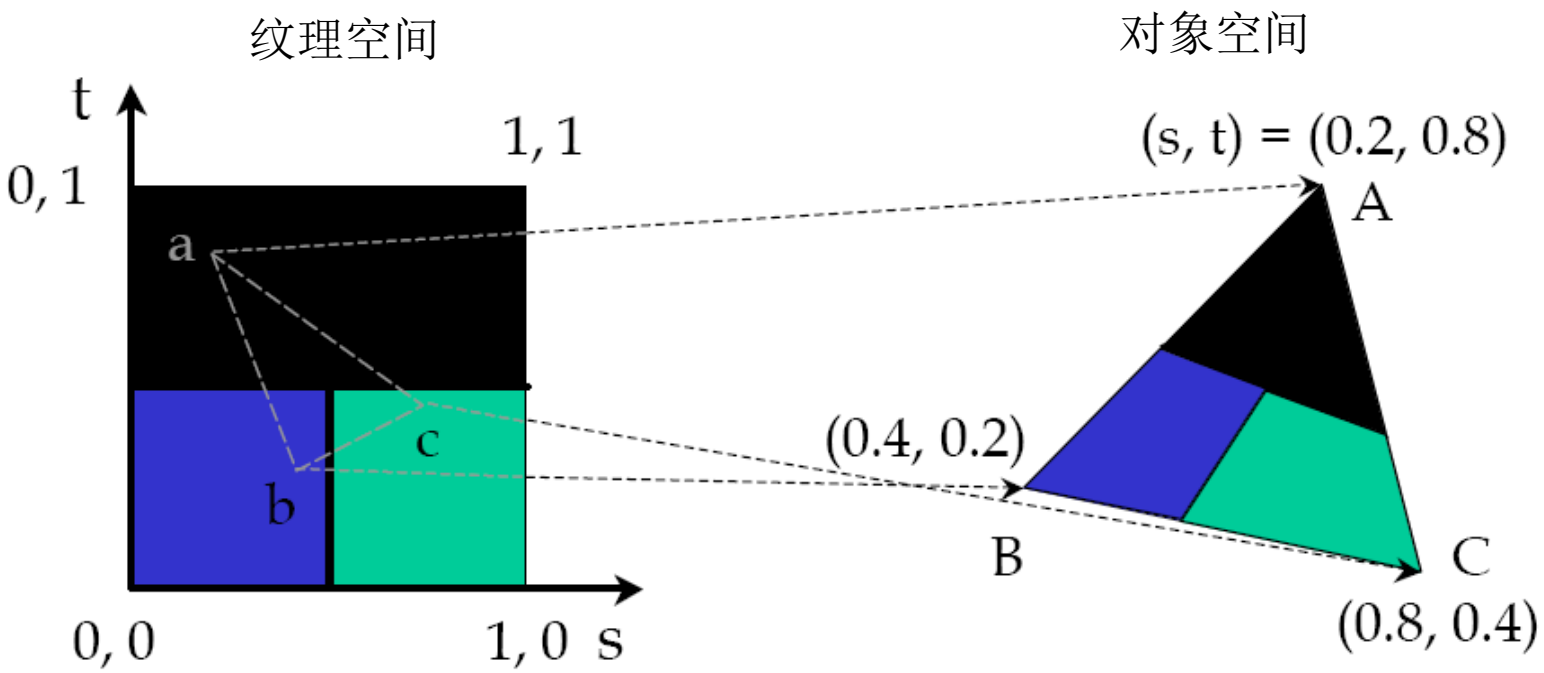
使用纹理

- 使用着色器从纹理对象中读取数据
  - 在着色器中定义采样器变量
    - `uniform sampler2D uTexture;`
  - 使用GLSL内置的Texture查询函数
    - `gvec4 texture(gsampler2D tex, vec2 P[, float bias]);`

- 多重纹理：OpenGL支持多重纹理
  - 纹理单元：`GL_TEXTURE0 – GL_TEXTUREi`
  - 激活纹理单元：`void glActiveTexture(GLenum texture);`
  - 设置纹理单元与采样器变量值之间的对应关系：在应用程序部分，uniform采样器看起来像uniform整数，使用glGetUniformLocation（或glGetActiveUniform）+ glUniform1i来设置

纹理坐标

- 基于参数纹理坐标
- 指定每个顶点对应的纹理坐标



纹理参数

- OpenGL中有许多办法确定纹理的使用方式
  - Wrapping参数确定当s, t的值超出[0,1]区间后的处理方法
  - 应用filter模式就会不采用点取样方法，而是采用区域平均方法
  - Mimmapping技术使得能以不同的分辨率应用纹理
  - 环境参数确定纹理映射与明暗处理的交互作用

Wrapping模式

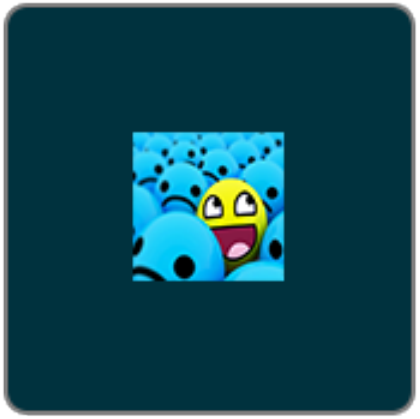
- 截断：若s,t>1就取1，若s,t<0就取0
- 重复：应用s,t模1的值

```
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_S,GL_REPEAT)
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_WRAP_T,GL_REPEAT)
```

GL_REPEAT

GL_MIRRORED_REPEAT

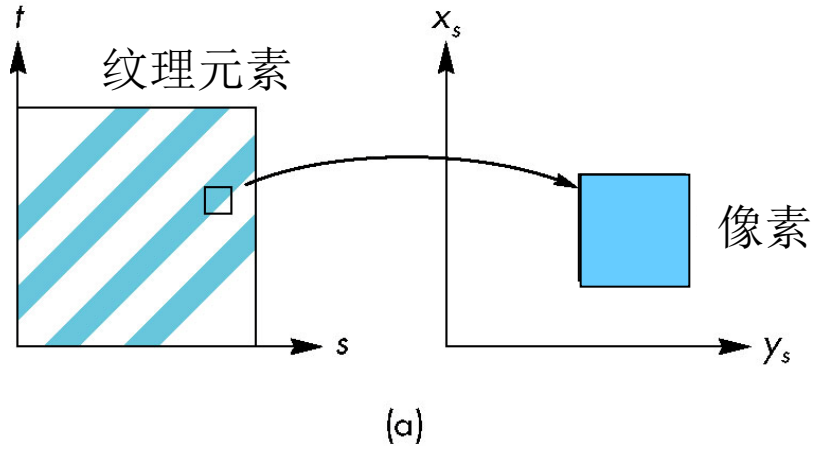
GL_CLAMP_TO_EDGE



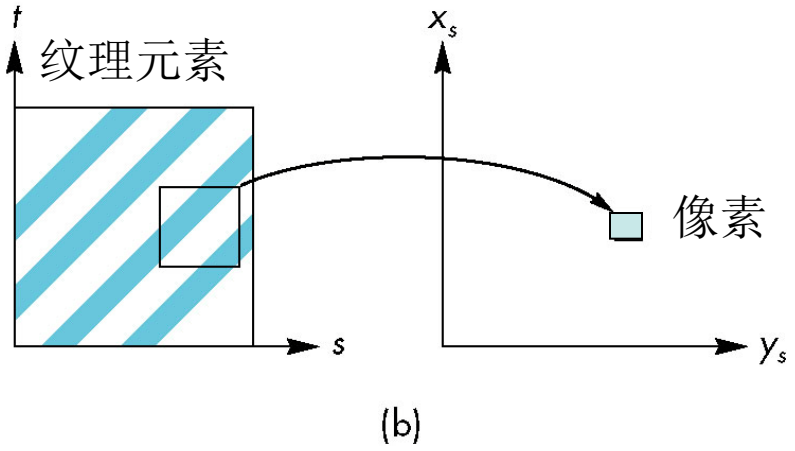
GL_CLAMP_TO_BORDER

放大与缩小

- 可以是多个纹理元素覆盖一个像素（缩小），也可以是多个像素覆盖一个纹理元素（放大）



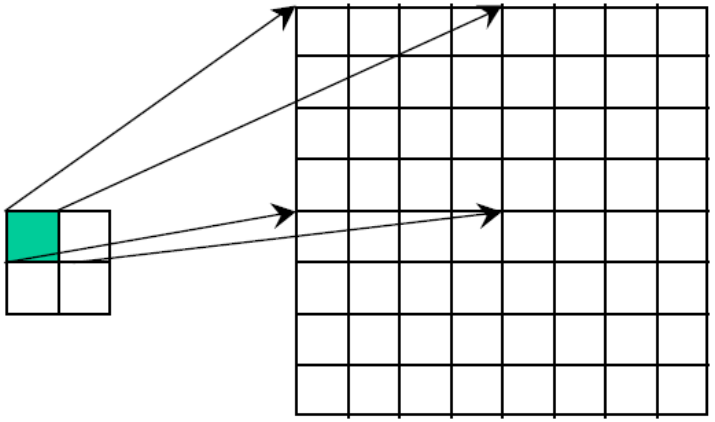
(a)



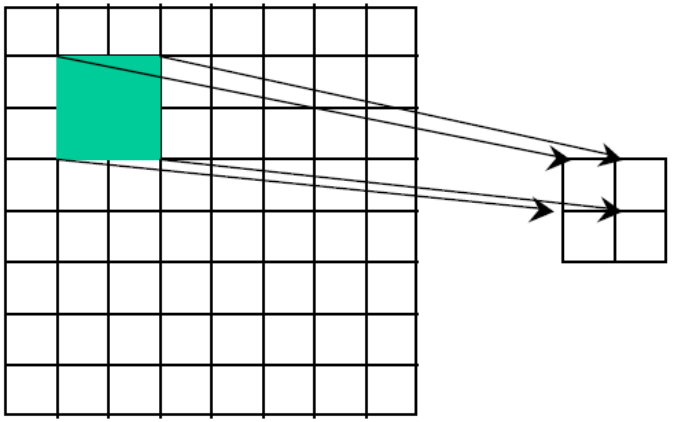
(b)

解决方法

- 可以应用点取样（最近纹理元素）或者线性滤波（2x2滤波）得到纹理值



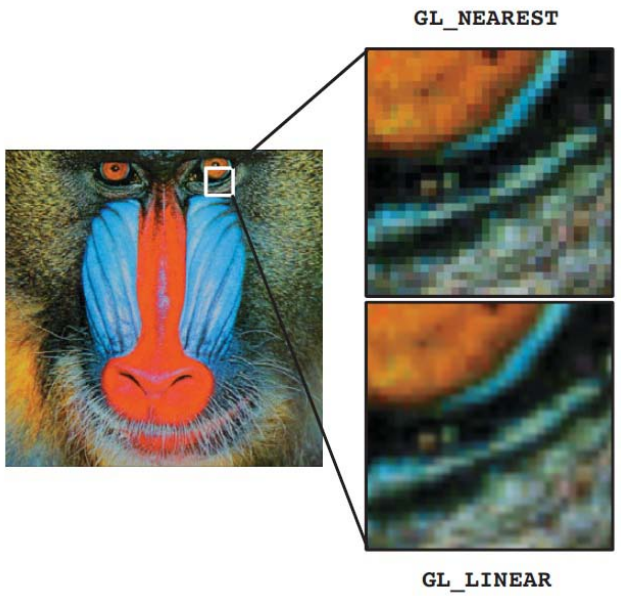
放大



缩小

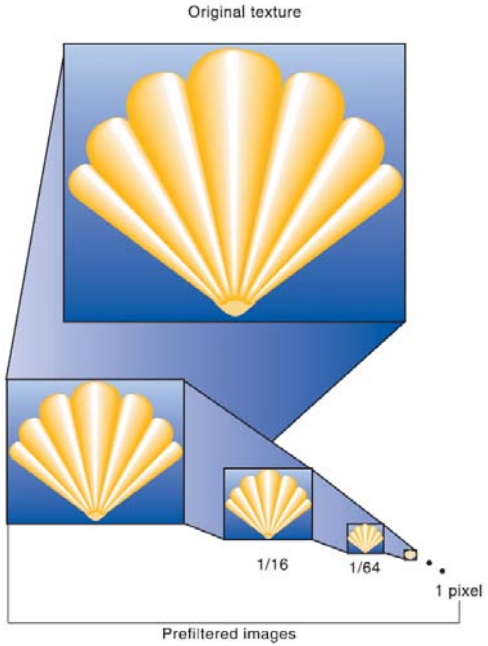
滤波模式

- 模式指定
  - `glTexParameteri(target, type, mode)`
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);`
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);`
- 注意在线性滤波中为滤波边界需要纹理元素具有额外的边界（border=1）

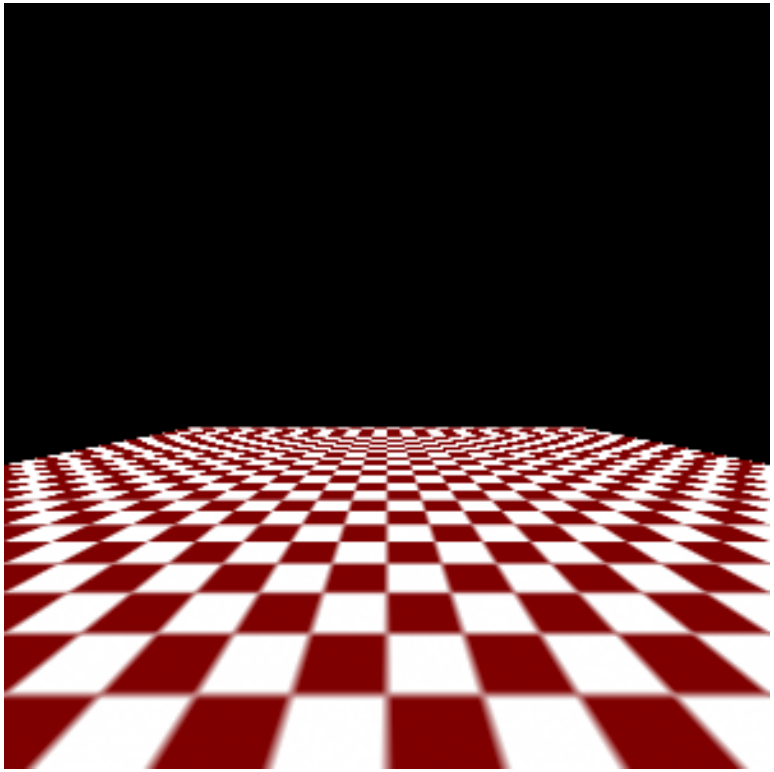


纹理的Mipmap

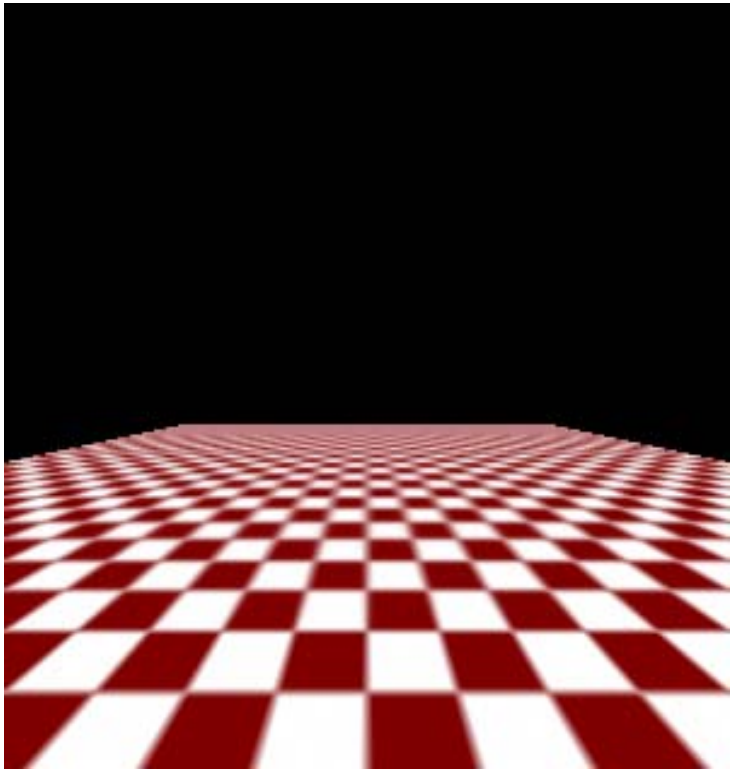
- Mipmap对纹理位图进行预先滤波，降低分辨率
- 可以减小对于非常小的要加纹理的对象的插值误差
- 在纹理定义时声明mipmap的层次
  - `glTexImage2D(GL_TEXTURE_*D, level, …);`
- 或者让OpenGL从给定图像建立起所有的mipmap纹理
  - `void glGenerateMipmap(GLenum target);`
  - `void glGenerateTextureMipmap(GLuint texture);`



有无mipmap的对比



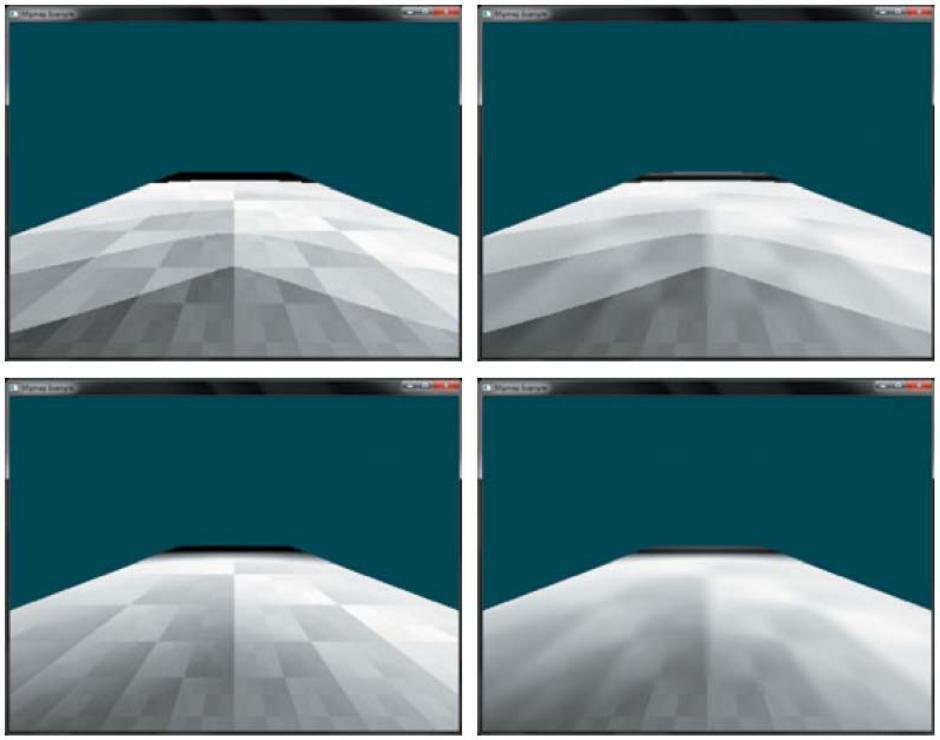
无mipmap



有mipmap

示例

GL_NEAREST_MIPMAP_NEAREST



GL_LINEAR_MIPMAP_NEAREST

GL_NEAREST_MIPMAP_LINEAR

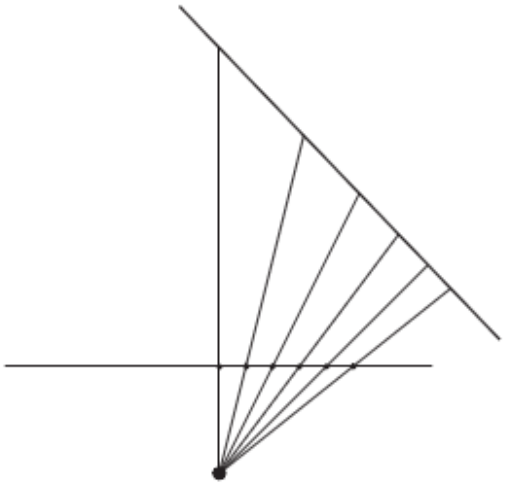
GL_LINEAR_MIPMAP_LINEAR

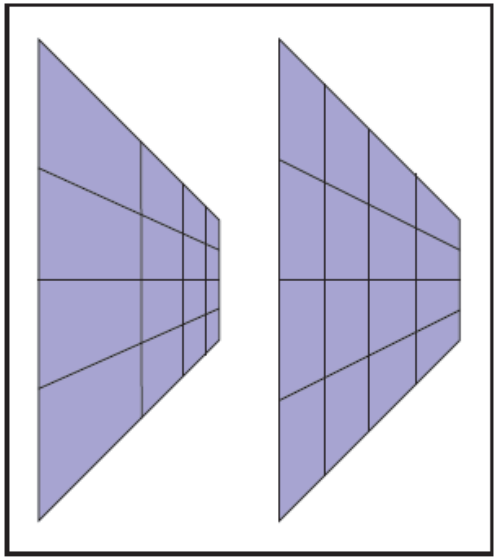
透视校正

- 透视正确的插值（Perspective-Correct Interpolation）
- 屏幕坐标空间插值 vs. 世界坐标插值

$$t = \frac{(\mathbf{p}_r - \mathbf{p}_a) \cdot (\mathbf{p}_b - \mathbf{p}_a)}{\|\mathbf{p}_b - \mathbf{p}_a\|^2}.$$

$$f = \frac{(1-t) f_a/w_a + t f_b/w_b}{(1-t)/w_a + t/w_b}$$





- 在着色器中，通过关键字
  - flat：不插值
  - noperspective：屏幕坐标空间插值
  - smooth：透视正确的插值，缺省的方式

Thanks for your attention!